Advanced Data Structures and Algorithm Analysis

丁尧相

浙江大学

Fall and Winter 2025

Lecture 8

Inverted File Index

- Inverted File Index
- Take-home messages

Inverted File Index

- Inverted File Index
- Take-home messages

AVL Trees

AVL Trees

Splay Trees

AVL Trees

Splay Trees

Red-Black Trees

AVL Trees

Splay Trees

Red-Black Trees

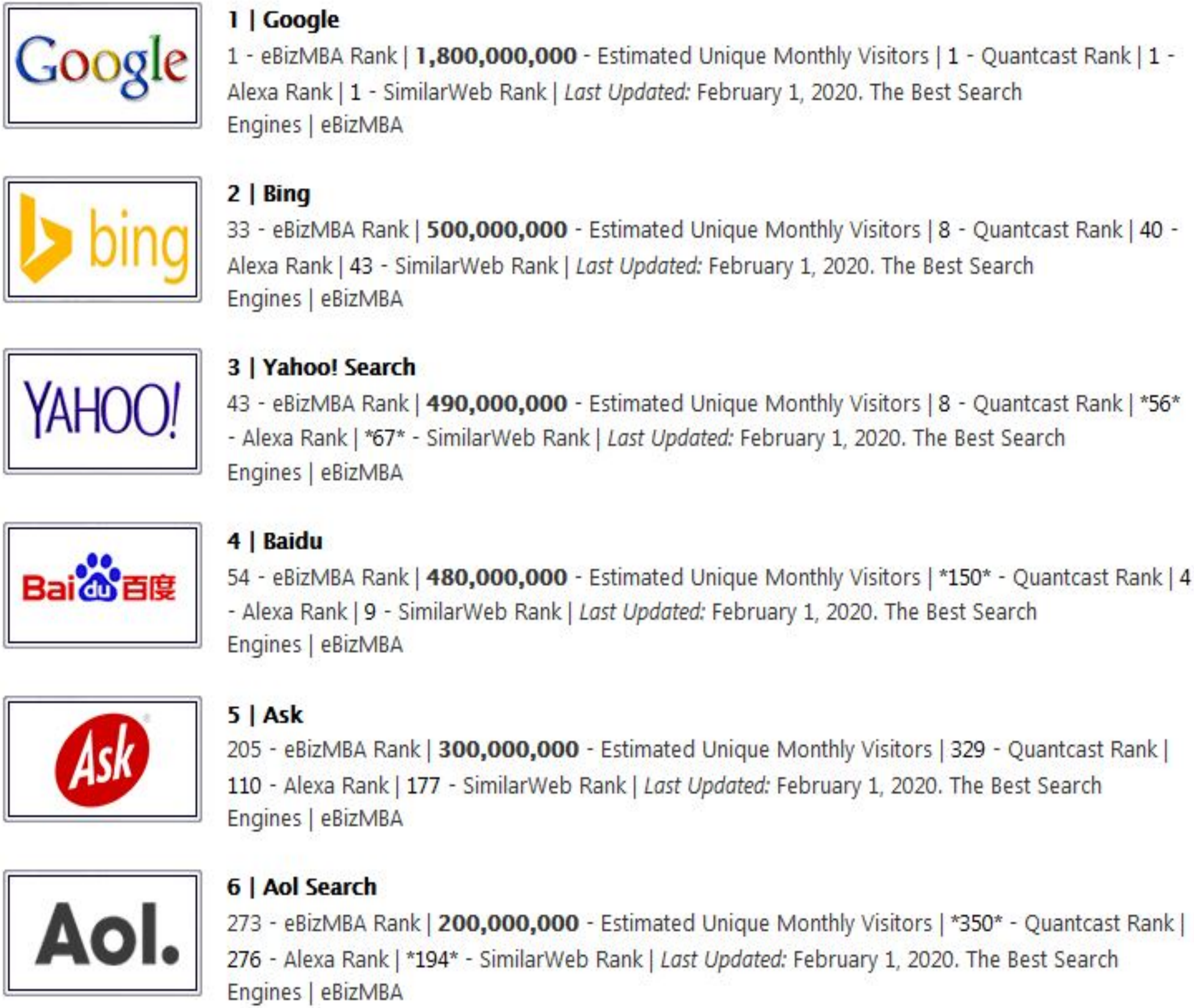
B+ Trees

AVL Trees

Splay Trees

Red-Black Trees

B+ Trees



1 | Google

1 - eBizMBA Rank | **1,800,000,000** - Estimated Unique Monthly Visitors | 1 - Quantcast Rank | 1 - Alexa Rank | 1 - SimilarWeb Rank | *Last Updated:* February 1, 2020. The Best Search Engines | eBizMBA

2 | Bing

33 - eBizMBA Rank | **500,000,000** - Estimated Unique Monthly Visitors | 8 - Quantcast Rank | 40 - Alexa Rank | 43 - SimilarWeb Rank | *Last Updated:* February 1, 2020. The Best Search Engines | eBizMBA

3 | Yahoo! Search

43 - eBizMBA Rank | **490,000,000** - Estimated Unique Monthly Visitors | 8 - Quantcast Rank | *56* - Alexa Rank | *67* - SimilarWeb Rank | *Last Updated:* February 1, 2020. The Best Search Engines | eBizMBA

4 | Baidu

54 - eBizMBA Rank | **480,000,000** - Estimated Unique Monthly Visitors | *150* - Quantcast Rank | 4 - Alexa Rank | 9 - SimilarWeb Rank | *Last Updated:* February 1, 2020. The Best Search Engines | eBizMBA

5 | Ask

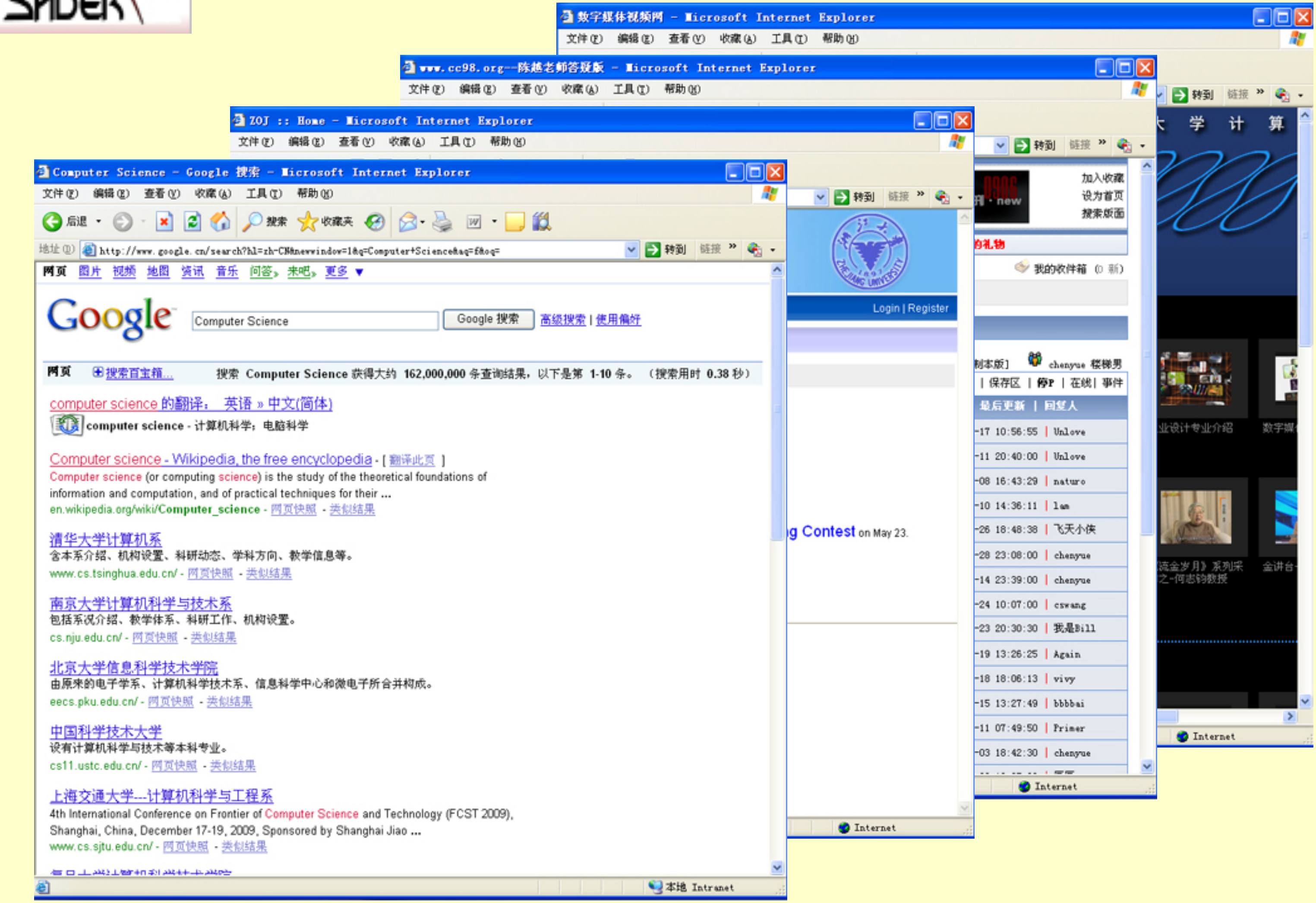
205 - eBizMBA Rank | **300,000,000** - Estimated Unique Monthly Visitors | 329 - Quantcast Rank | 110 - Alexa Rank | 177 - SimilarWeb Rank | *Last Updated:* February 1, 2020. The Best Search Engines | eBizMBA

6 | Aol Search

273 - eBizMBA Rank | **200,000,000** - Estimated Unique Monthly Visitors | *350* - Quantcast Rank | 276 - Alexa Rank | *194* - SimilarWeb Rank | *Last Updated:* February 1, 2020. The Best Search Engines | eBizMBA

Inverted File Index

Inverted File Index


数字媒体视频网 - Microsoft Internet Explorer
www.cc98.org--陈越老师答疑版 - Microsoft Internet Explorer
ZOJ :: Home - Microsoft Internet Explorer
Computer Science - Google 搜索 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D)
转到 链接
Login | Register
g Contest on May 23.
加入收藏
设为首页
搜索版面
我的收件箱 (0 新)
保存区 | 停P | 在线 | 事件
最后更新 | 回复人
大 学 计 算
Internet
本地 Intranet

http://www.google.cn/search?hl=zh-CN&newwindow=1&q=Computer+Science&aq=f&oq=

网页 图片 视频 地图 资讯 音乐 问答» 来吧» 更多 ▼

Computer Science | Google 搜索 | 高级搜索 | 使用偏好

网页 ⊞搜索百宝箱... 搜索 Computer Science 获得大约 162,000,000 条查询结果，以下是第 1-10 条。（搜索用时 0.38 秒）

computer science 的翻译： 英语 » 中文(简体)
computer science - 计算机科学；电脑科学

Computer science - Wikipedia, the free encyclopedia - [翻译此页]
Computer science (or computing science) is the study of the theoretical foundations of information and computation, and of practical techniques for their ...
en.wikipedia.org/wiki/Computer_science - 网页快照 - 类似结果

清华大学计算机系
含本系介绍、机构设置、科研动态、学科方向、教学信息等。
www.cs.tsinghua.edu.cn/ - 网页快照 - 类似结果

南京大学计算机科学与技术系
包括系况介绍、教学体系、科研工作、机构设置。
cs.nju.edu.cn/ - 网页快照 - 类似结果

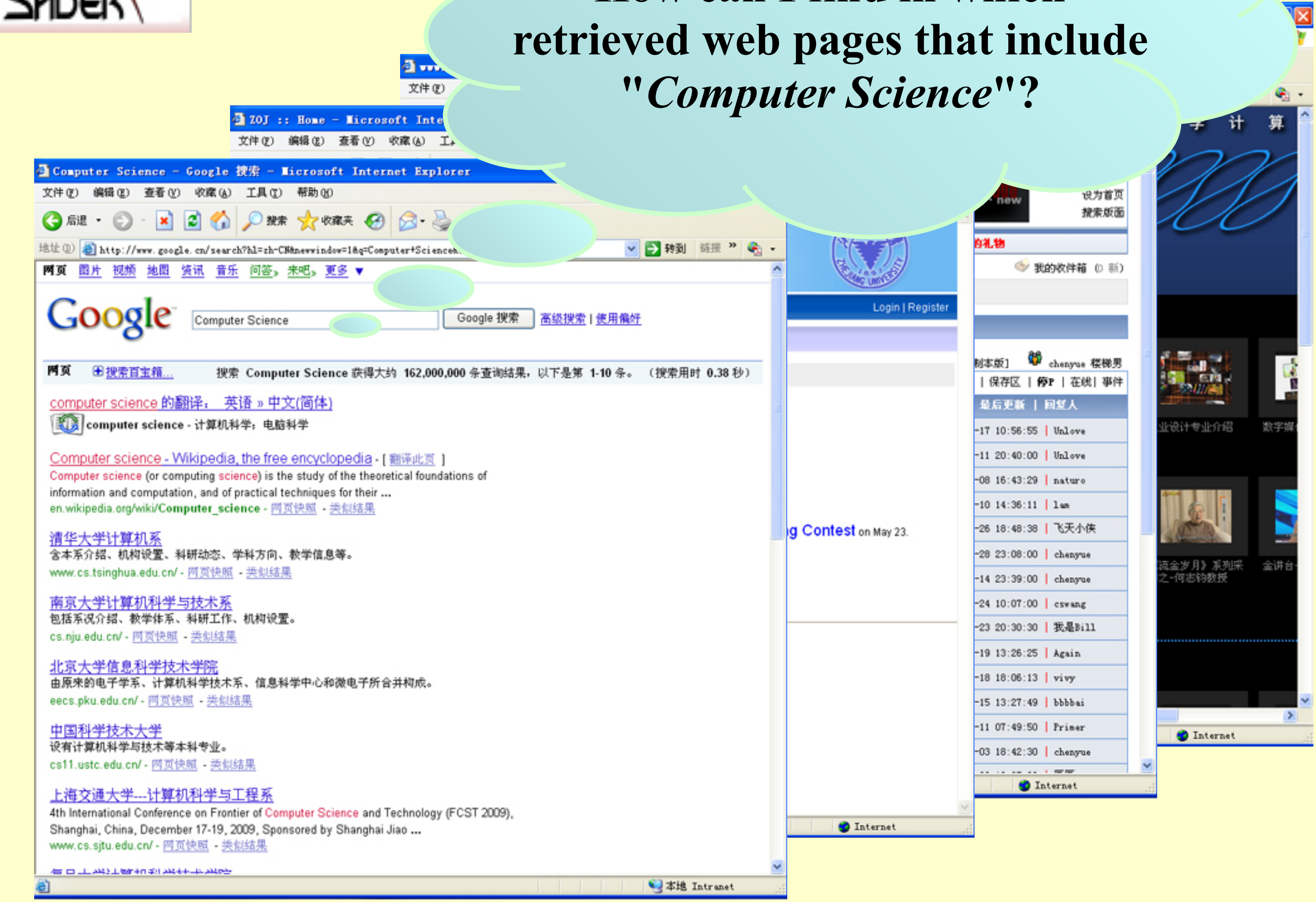
北京大学信息科学技术学院
由原来的电子学系、计算机科学技术系、信息科学中心和微电子所合并构成。
eecs.pku.edu.cn/ - 网页快照 - 类似结果

中国科学技术大学
设有计算机科学与技术等本科专业。
cs11.ustc.edu.cn/ - 网页快照 - 类似结果

上海交通大学---计算机科学与工程系
4th International Conference on Frontier of Computer Science and Technology (FCST 2009), Shanghai, China, December 17-19, 2009, Sponsored by Shanghai Jiao ...
www.cs.sjtu.edu.cn/ - 网页快照 - 类似结果

Inverted File Index

How can I find in which retrieved web pages that include "*Computer Science*"?


Computer Science - Google 搜索 - Microsoft Internet Explorer
ZOJ :: Home - Microsoft Inte
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://www.google.cn/search?hl=zh-CN&newwindow=1&q=Computer+Science
转到 链接
网页 图片 视频 地图 资讯 音乐 问答 来吧 更多
Google
Google 搜索
高级搜索 | 使用偏好
Login | Register
设为首页
搜索版面
我的收件箱
保存区 | 在线 | 事件
最后更新 | 回复人
计 算
本地 Intranet
Internet

Computer Science

网页 搜索百宝箱... 搜索 Computer Science 获得大约 162,000,000 条查询结果，以下是第 1-10 条。（搜索用时 0.38 秒）

computer science 的翻译： 英语 » 中文(简体)
computer science - 计算机科学；电脑科学

Computer science - Wikipedia, the free encyclopedia - [翻译此页]
Computer science (or computing science) is the study of the theoretical foundations of information and computation, and of practical techniques for their ...
en.wikipedia.org/wiki/Computer_science - 网页快照 - 类似结果

清华大学计算机系
含本系介绍、机构设置、科研动态、学科方向、教学信息等。
www.cs.tsinghua.edu.cn/ - 网页快照 - 类似结果

南京大学计算机科学与技术系
包括系况介绍、教学体系、科研工作、机构设置。
cs.nju.edu.cn/ - 网页快照 - 类似结果

北京大学信息科学技术学院
由原来的电子学系、计算机科学技术系、信息科学中心和微电子所合并构成。
eecs.pku.edu.cn/ - 网页快照 - 类似结果

中国科学技术大学
设有计算机科学与技术等本科专业。
cs11.ustc.edu.cn/ - 网页快照 - 类似结果

上海交通大学---计算机科学与工程系
4th International Conference on Frontier of Computer Science and Technology (FCST 2009), Shanghai, China, December 17-19, 2009, Sponsored by Shanghai Jiao ...
www.cs.sjtu.edu.cn/ - 网页快照 - 类似结果

g Contest on May 23.

-17 10:56:55	Unlove
-11 20:40:00	Unlove
-08 16:43:29	naturo
-10 14:36:11	lan
-26 18:48:38	飞天小侠
-28 23:08:00	chenyue
-14 23:39:00	chenyue
-24 10:07:00	cswang
-23 20:30:30	我是Bill
-19 13:26:25	Again
-18 18:06:13	vivy
-15 13:27:49	bbbbai
-11 07:49:50	Primer
-03 18:42:30	chenyue

Information Retrieval

Information Retrieval

- Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Information Retrieval

- Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

  - These days we frequently think first of web search, but there are many other cases:
    - E-mail search
    - Searching your laptop
    - Corporate knowledge bases
    - Legal information retrieval

Basic assumptions of Information Retrieval

Basic assumptions of Information Retrieval

- Collection: A set of documents
  - Assume it is a static collection for the moment

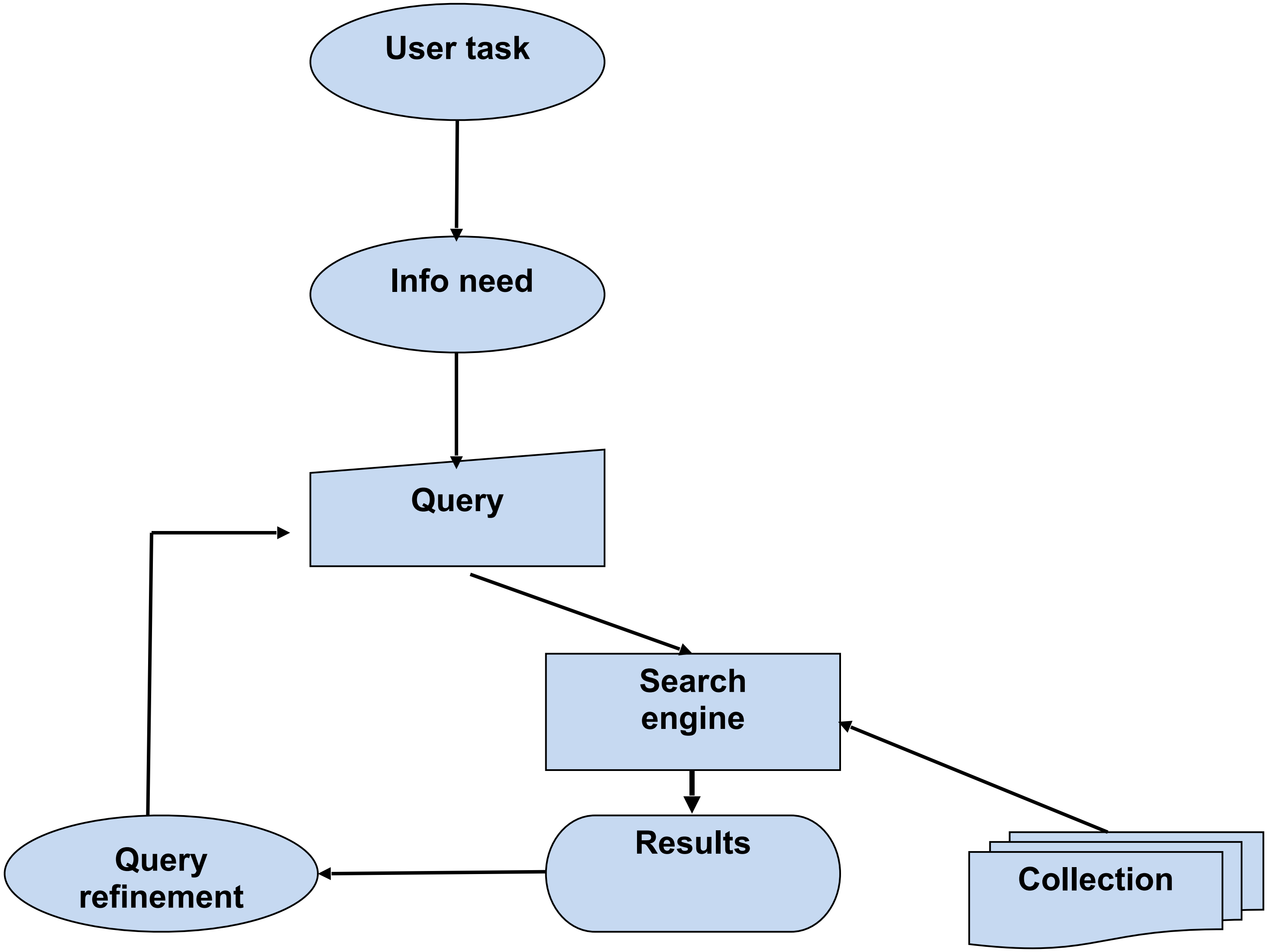
Basic assumptions of Information Retrieval

- Collection: A set of documents
  - Assume it is a static collection for the moment

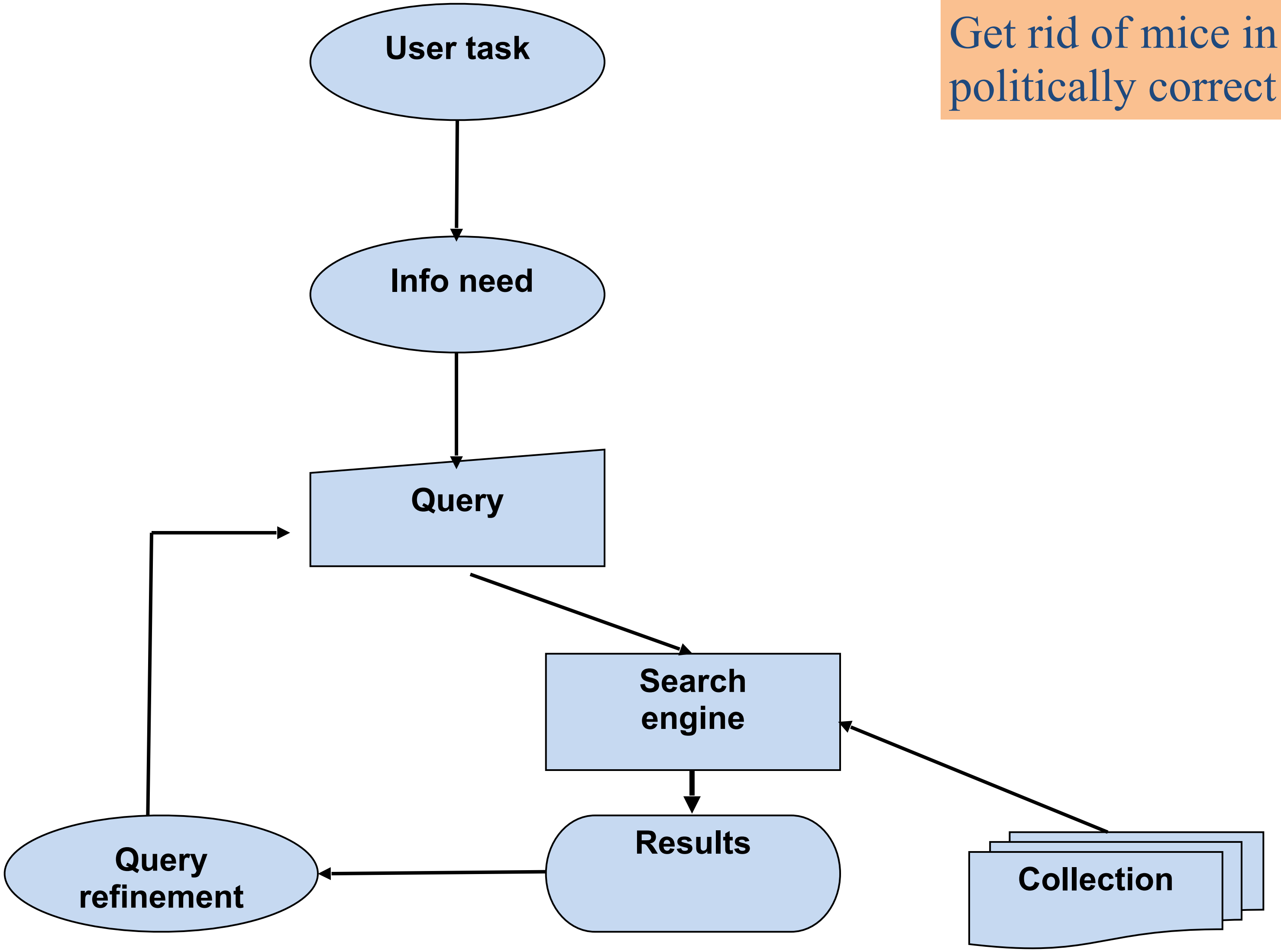
- Goal: Retrieve documents with information that is relevant to the user's information need and helps the user complete a task

This is different from searching for a fixed key in a data structure or database

The classic search model

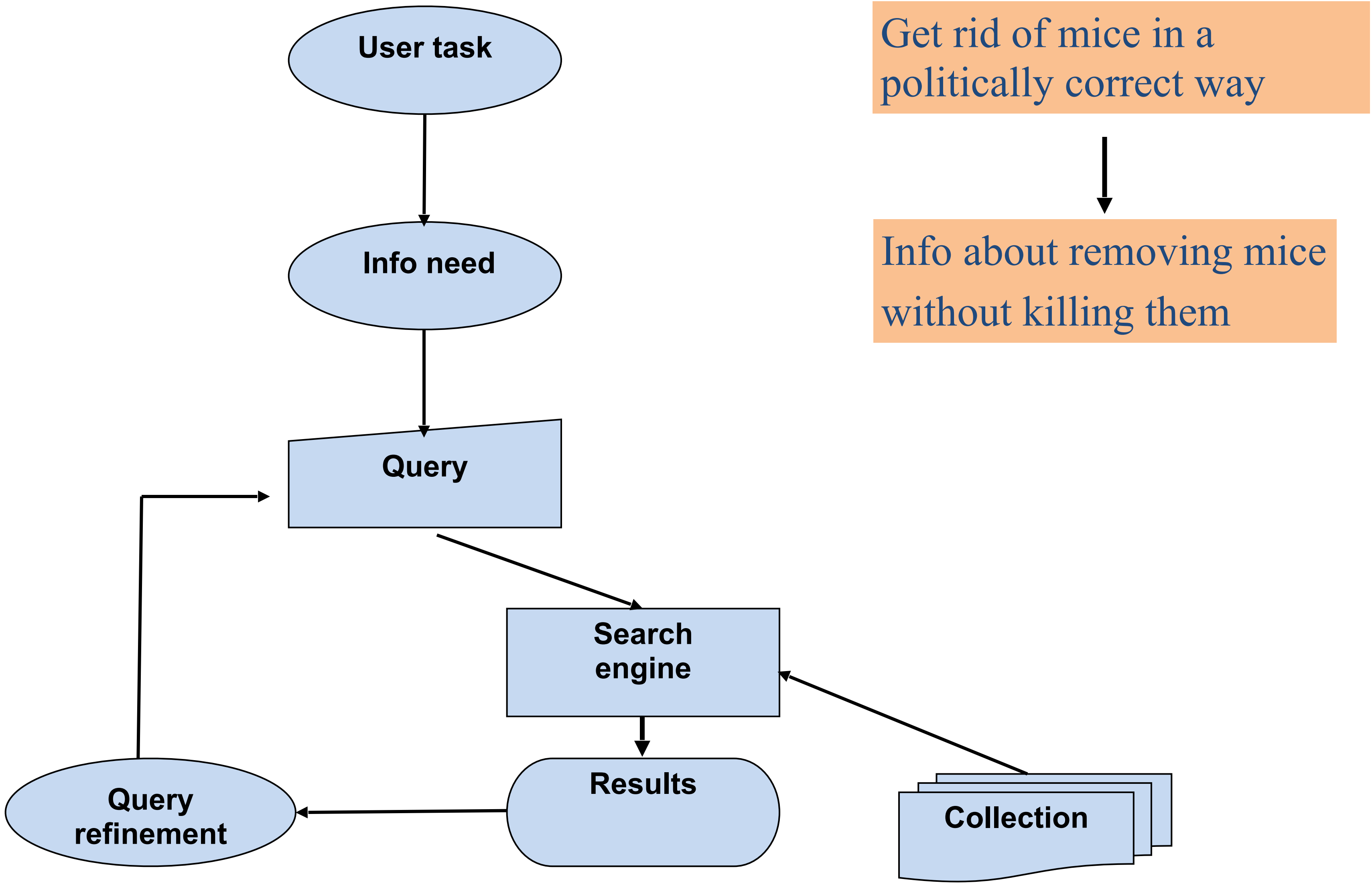


The classic search model



Get rid of mice in a politically correct way

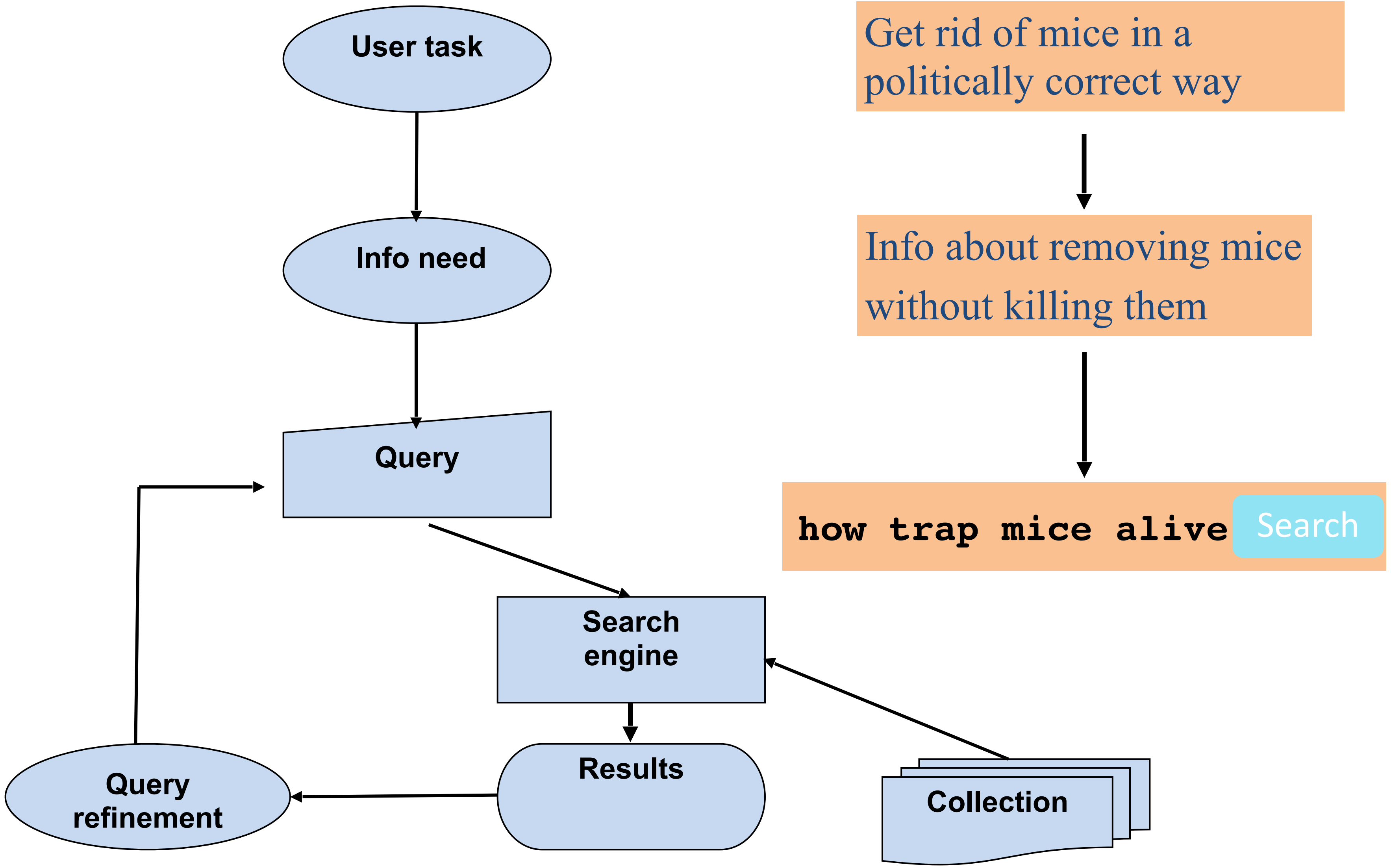
The classic search model



Get rid of mice in a politically correct way

Info about removing mice without killing them

The classic search model



User task

Info need

Query

Search engine

Results

Query refinement

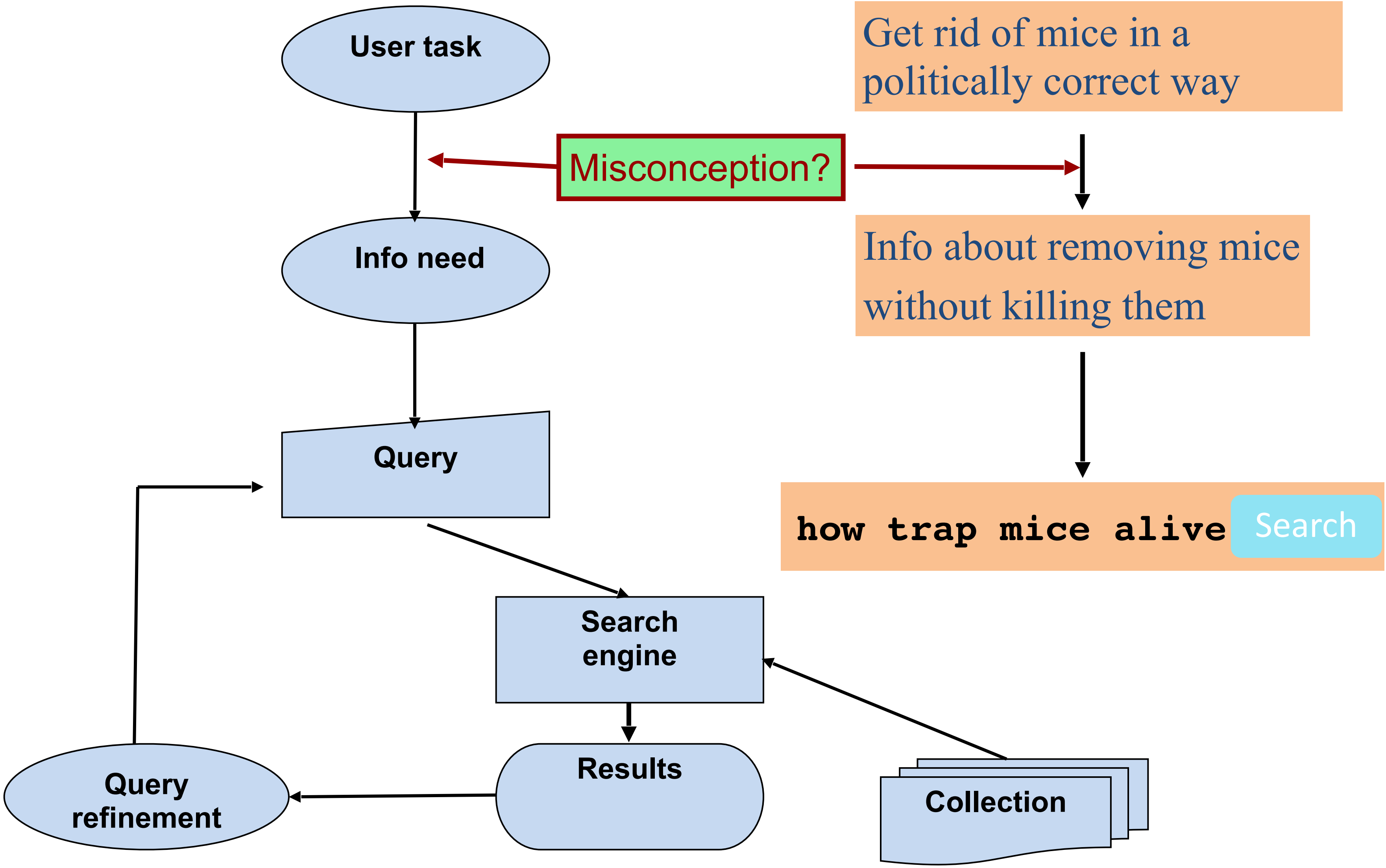
Collection

Get rid of mice in a politically correct way

Info about removing mice without killing them

how trap mice alive

The classic search model

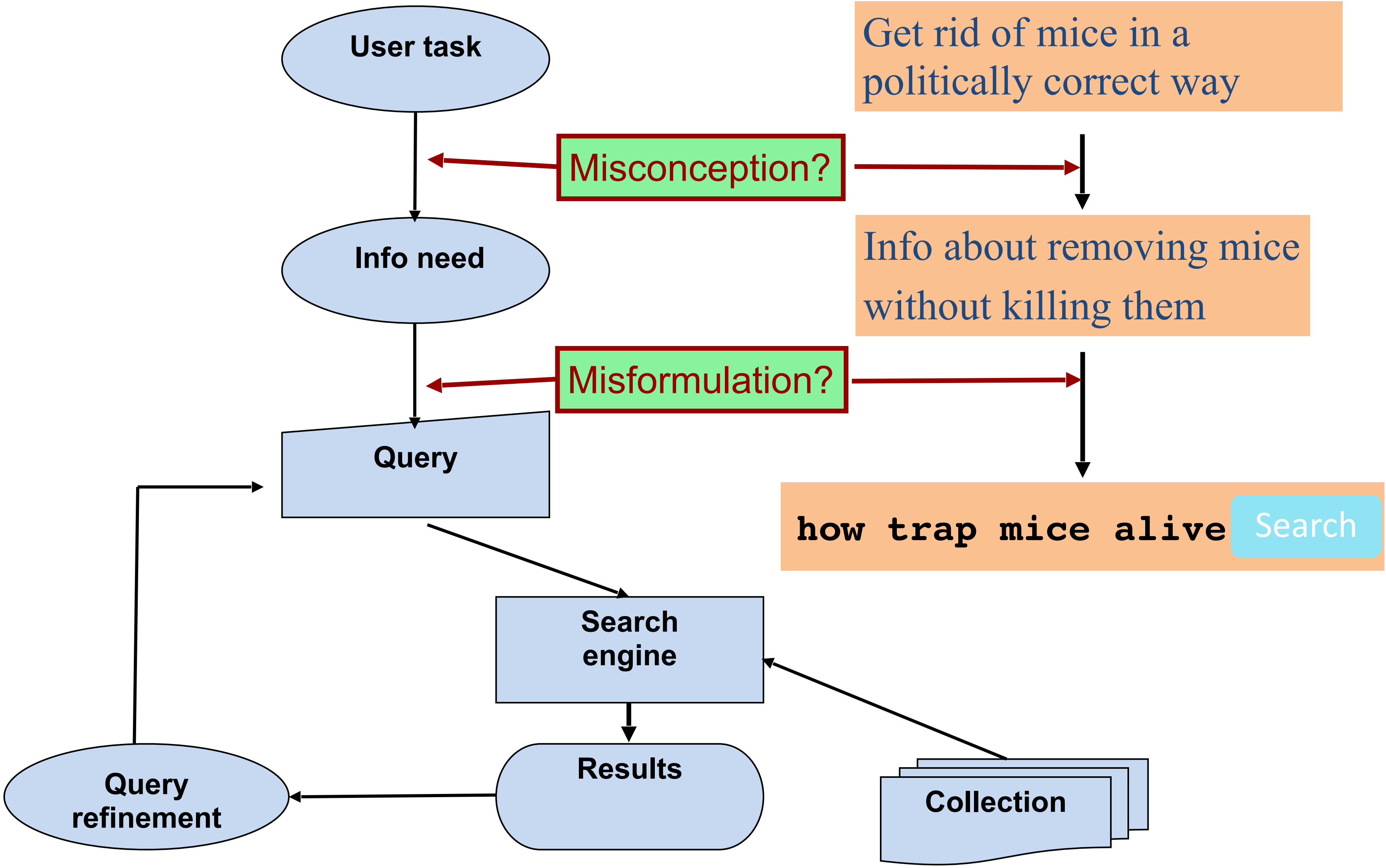


Get rid of mice in a politically correct way

Info about removing mice without killing them

`how trap mice alive` Search

The classic search model



User task

Info need

Query

Search engine

Results

Query refinement

Collection

Misconception?

Misformulation?

Get rid of mice in a politically correct way

Info about removing mice without killing them

`how trap mice alive` Search

Inverted File Index

☞ **Solution 1:** **Scan each page for the string "Computer Science".**

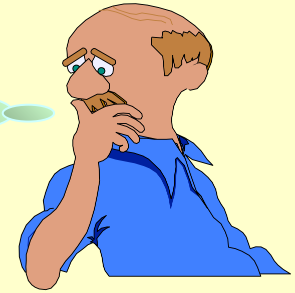
☞ **Solution 1:** **Scan each page for the string "Computer Science".**

Wait till your next life !

☞ **Solution 1:** **Scan each page for the string "Computer Science".**

Wait till your next life !

How did Google do?



Inverted File Index

☞ **Solution 1:** **Scan each page for the string "Computer Science".**

Wait till your next life !

Have more than 1 billion web pages Indexed

☞ **Solution 1:** **Scan each page for the string "Computer Science".**

Wait till your next life !

Have more than 1 trillion web pages Indexed

☞ **Solution 2: Term-Document Incidence Matrix**

☞ **Solution 2: Term-Document Incidence Matrix**

〖Example〗 Document sets

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

☞ Solution 2: Term-Document Incidence Matrix

〖Example〗 Document sets

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

	1	2	3	4
a	0	1	1	1
arrived	0	0	1	1
damaged	0	1	0	0
delivery	0	0	1	0
fire	0	1	0	0
gold	1	1	0	1
of	0	1	1	1
in	0	1	1	1
shipment	0	1	0	1
silver	1	0	1	0
truck	1	0	1	1

☞ Solution 2: Term-Document Incidence Matrix

〖Example〗 Document sets

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

	1	2	3	4
a	0	1	1	1
arrived	0	0	1	1
damaged	0	1	0	0
delivery	0	0	1	0
fire	0	1	0	0
gold	1	1	0	1
of	0	1	1	1
in	0	1	1	1
shipment	0	1	0	1
silver	1	0	1	0
truck	1	0	1	1

silver & truck

☞ Solution 2: Term-Document Incidence Matrix

〖Example〗 Document sets

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

	1	2	3	4
a	0	1	1	1
arrived	0	0	1	1
damaged	0	1	0	0
delivery	0	0	1	0
fire	0	1	0	0
gold	1	1	0	1
of	0	1	1	1
in	0	1	1	1
shipment	0	1	0	1
silver	1	0	1	0
truck	1	0	1	1

silver & truck = 1010 & 1011 = 1010

☞ **Solution 2: Term-Document Incidence Matrix**

〖Example〗 **Document sets**

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

	1	2	3	4
a	0	1	1	1
arrived	0	0	1	1
damaged	0	1	0	0
delivery	0	0	1	0
fire	0	1	0	0
gold	1	1	0	1
of	0	1	1	1
in	0	1	1	1
shipment	0	1	0	1
silver	1	0	1	0
truck	1	0	1	1

silver & truck = 1010 & 1011 = 1010

☞ **Solution 3: Compact Version - Inverted File Index**

☞ **Solution 3: Compact Version - Inverted File Index**

【Definition】 Index is a mechanism for locating a given term in a text.

☞ **Solution 3: Compact Version - Inverted File Index**

【Definition】 **Index is a mechanism for locating a given term in a text.**

【Definition】 **Inverted file contains a list of pointers (e.g. the number of a page) to all occurrences of that term in the text.**

☞ Solution 3: Compact Version - Inverted File Index

【Definition】 Index is a mechanism for locating a given term in a text.

【Definition】 Inverted file contains a list of pointers (e.g. the number of a page) to all occurrences of that term in the text.

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

☞ Solution 3: Compact Version - Inverted File Index

【Definition】 Index is a mechanism for locating a given term in a text.

【Definition】 Inverted file contains a list of pointers (e.g. the number of a page) to all occurrences of that term in the text.

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

Inverted File Index

No.	Term	Times; Documents
1	a	<3; 2,3,4>
2	arrived	<2; 3,4>
3	damaged	<1; 2>
4	delivery	<1; 3>
5	fire	<1; 2>
6	gold	<3; 1,2,4>
7	of	<3; 2,3,4>
8	in	<3; 2,3,4>
9	shipment	<2; 2,4>
10	silver	<2; 1,3>
11	truck	<3; 1,3,4>

☞ **Solution 3: Compact Version - Inverted File Index**

【Definition】 **Index** is a mechanism for locating a given term in a text.

【Definition】 **Inverted file** contains a list of pointers (e.g. the number of a page) to all occurrences of that term in the text.

Inverted because it lists for a *term*, all documents that contain the term

Doc	Text
1	Gold silv
2	Shipment of go damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

Index

No.	Term	Times; Documents
[illegible]	[illegible]	[illegible] 2,3,4>
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	<1; 3>
5	fire	<1; 2>
6	gold	<3; 1,2,4>
7	of	<3; 2,3,4>
8	in	<3; 2,3,4>
9	shipment	<2; 2,4>
10	silver	<2; 1,3>
11	truck	<3; 1,3,4>

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents
1	a	<3; 2,3,4>
2	arrived	<2; 3,4>
3	damaged	<1; 2>
4	delivery	<1; 3>
5	fire	<1; 2>
6	gold	<3; 1,2,4>
7	of	<3; 2,3,4>
8	in	<3; 2,3,4>
9	shipment	<2; 2,4>
10	silver	<2; 1,3>
11	truck	<3; 1,3,4>

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

→

No.	Term	Times; Documents
1	a	<3; 2,3,4>
2	arrived	<2; 3,4>
3	damaged	<1; 2>
4	delivery	<1; 3>
5	fire	<1; 2>
6	gold	<3; 1,2,4>
7	of	<3; 2,3,4>
8	in	<3; 2,3,4>
9	shipment	<2; 2,4>
10	silver	<2; 1,3>
11	truck	<3; 1,3,4>

How to easily print the sentences which contain the words and highlight the words?

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents Words
1	a	<3; (2;6),(3;6),(4;6)>
2	arrived	<2; (3;4),(4;4)>
3	damaged	<1; (2;4)>
4	delivery	<1; (3;1)>
5	fire	<1; (2;7)>
6	gold	<3; (1;1),(2;3),(4;3)>
7	of	<3; (2;2),(3;2),(4;2)>
8	in	<3; (2;5),(3;5),(4;5)>
9	shipment	<2; (2;1),(4;1)>
10	silver	<2; (1;2),(3;3,7)>
11	truck	<3; (1;3),(3;8),(4;7)>

How to easily print the sentences which contain the words and highlight the words?

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents Words
1	a	<3; (2;6),(3;6),(4;6)>
2	arrived	<2; (3;4),(4;4)>
3	damaged	<1; (2;4)>
4	delivery	<1; (3;1)>
5	fire	<1; (2;7)>
6	gold	<3; (1;1),(2;3),(4;3)>
7	of	<3; (2;2),(3;2),(4;2)>
8	in	<3; (2;5),(3;5),(4;5)>
9	shipment	<2; (2;1),(4;1)>
10	silver	<2; (1;2),(3;3,7)>
11	truck	<3; (1;3),(3;8),(4;7)>

Term Dictionary

How to easily print the sentences which contain the words and highlight the words?

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents Words
1	a	<3; (2;6),(3;6),(4;6)>
2	arrived	<2; (3;4),(4;4)>
3	damaged	<1; (2;4)>
4	delivery	<1; (3;1)>
5	fire	<1; (2;7)>
6	gold	<3; (1;1),(2;3),(4;3)>
7	of	<3; (2;2),(3;2),(4;2)>
8	in	<3; (2;5),(3;5),(4;5)>
9	shipment	<2; (2;1),(4;1)>
10	silver	<2; (1;2),(3;3,7)>
11	truck	<3; (1;3),(3;8),(4;7)>

Term Dictionary

Posting List

How to easily print the sentences which contain the words and highlight the words?

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents Words
1	a	<3; (2;6),(3;6),(4;6)>
2	arrived	<2; (3;4),(4;4)>
3	damaged	<1; (2;4)>
4	delivery	<1; (3;1)>
5	fire	<1; (2;7)>
6	gold	<3; (1;1),(2;3),(4;3)>
7	of	<3; (2;2),(3;2),(4;2)>
8	in	<3; (2;5),(3;5),(4;5)>
9	shipment	<2; (2;1),(4;1)>
10	silver	<2; (1;2),(3;3,7)>
11	truck	<3; (1;3),(3;8),(4;7)>

Term Dictionary

Posting List

How to easily print the sentences which contain the words and highlight the words?

Why do we keep “times” (frequency)?

Index Generator

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Vocabulary
Scanner

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Vocabulary
Scanner

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Vocabulary
Scanner

Vocabulary
Insertor

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Vocabulary
Scanner

Vocabulary
Insertor

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Vocabulary
Scanner

Vocabulary
Insertor

Memory management

While reading a term ……

While reading a term ……

✂ ***Word Stemming***

While reading a term ……

✂ ***Word Stemming***

Process a word so that only its stem or root form is left.

Inverted File Index

While reading a term ……

✂ ***Word Stemming***

Process a word so that only its stem or root form is left.

〖Example〗

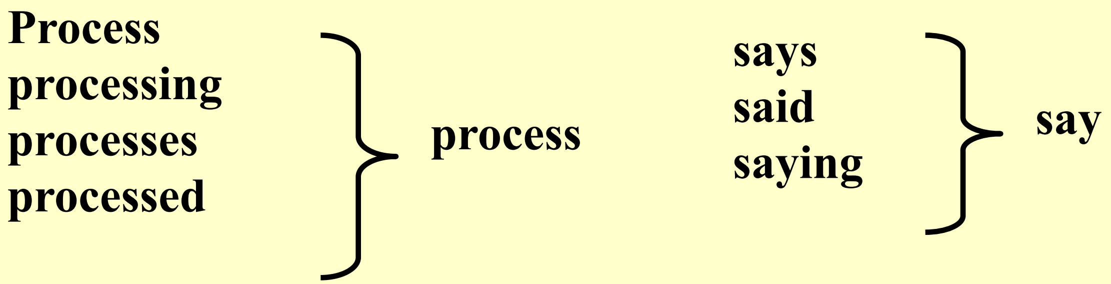
$$\left.\begin{array}{l}\textbf{Process}\\ \textbf{processing}\\ \textbf{processes}\\ \textbf{processed}\end{array}\right\}\ \textbf{process} \qquad \left.\begin{array}{l}\textbf{says}\\ \textbf{said}\\ \textbf{saying}\end{array}\right\}\ \textbf{say}$$

While reading a term ……

✂ ***Word Stemming***

Process a word so that only its stem or root form is left.

〖Example〗



Process
processing
processes
processed

process

says
said
saying

say

✂ ***Stop Words***

While reading a term ……

✂ ***Word Stemming***

Process a word so that only its stem or root form is left.

〖Example〗

Process
processing
processes
processed
} **process**

says
said
saying
} **say**

✂ ***Stop Words***

Some words are so common that almost every document contains them, such as "a" "the" "it". It is useless to index them. They are called *stop words*. We can eliminate them from the original documents.

While accessing a term ……

While accessing a term ……

☞ **Solution 1:** **Search trees (** ***B- trees, B+ trees, Tries, …*** **)**

While accessing a term ……

- ☞ **Solution 1:** **Search trees (*B- trees, B+ trees, Tries, …*)**

- ☞ **Solution 2:** **Hashing**

While accessing a term ……

- ☞ Solution 1: Search trees (*B- trees, B+ trees, Tries, ...*)

- ☞ Solution 2: Hashing

Discussion 3:
What are the pros and cons of using hashing, comparing to using search trees?

While accessing a term ……

- ☞ **Solution 1: Search trees (*B- trees, B+ trees, Tries, …*)**

- ☞ **Solution 2: Hashing**

Discussion 3:
What are the pros and cons of using hashing, comparing to using search trees?

- 👍 **faster for one word**
- 👎 **scanning in sequential order is not possible (e.g. range searches are expensive)**

While not having enough memory ……

While not having enough memory ……

```
while ( read a document D ) {
  while ( read a term T in D ) {
    if ( Find( Dictionary, T ) == false )
      Insert( Dictionary, T );
    Get T's posting list;
    Insert a node to T's posting list;
  }
}
for ( i=0; i<BlockCnt; i++ )
```

While not having enough memory ……

```
BlockCnt = 0;
while ( read a document D ) {
  while ( read a term T in D ) {
    if ( out of memory ) {
      Write BlockIndex[BlockCnt] to disk;
      BlockCnt ++;
      FreeMemory;
    }
    if ( Find( Dictionary, T ) == false )
      Insert( Dictionary, T );
    Get T's posting list;
    Insert a node to T's posting list;
  }
}
for ( i=0; i<BlockCnt; i++ )
```

While not having enough memory ……

```
BlockCnt = 0;
while ( read a document D ) {
  while ( read a term T in D ) {
    if ( out of memory ) {
      Write BlockIndex[BlockCnt] to disk;
      BlockCnt ++;
      FreeMemory;
    }
    if ( Find( Dictionary, T ) == false )
      Insert( Dictionary, T );
    Get T’s posting list;
    Insert a node to T’s posting list;
  }
}
for ( i=0; i<BlockCnt; i++ )
  Merge( InvertedIndex, BlockIndex[i] );
```

While not having enough memory ……

```
BlockCnt = 0;
while ( read a document D ) {
  while ( read a term T in D ) {
    if ( out of memory ) {
      Write BlockIndex[BlockCnt] to disk;
      BlockCnt ++;
      FreeMemory;
    }
    if ( Find( Dictionary, T ) == false )
      Insert( Dictionary, T );
    Get T's posting list;
    Insert a node to T's posting list;
  }
}
for ( i=0; i<BlockCnt; i++ )
  Merge( InvertedIndex, BlockIndex[i] );
```

Sorted

Distributed indexing (for web-scale indexing — don’t try this at home!)

Distributed indexing (for web-scale indexing — don't try this at home!)

—— Each node contains index of a subset of collection

Distributed indexing (for web-scale indexing — don't try this at home!)

—— Each node contains index of a subset of collection

Distributed indexing (for web-scale indexing — don’t try this at home!)

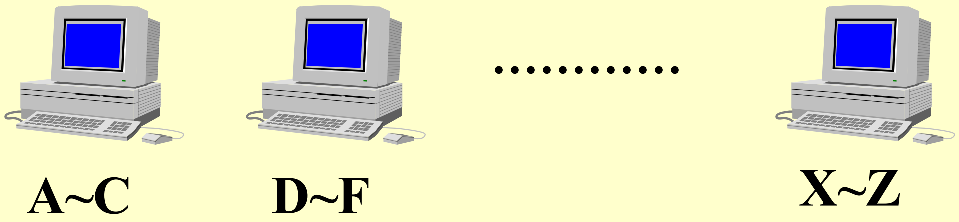
—— Each node contains index of a subset of collection

☞ **Solution 1: Term-partitioned index**

Distributed indexing (for web-scale indexing — don't try this at home!)

—— Each node contains index of a subset of collection

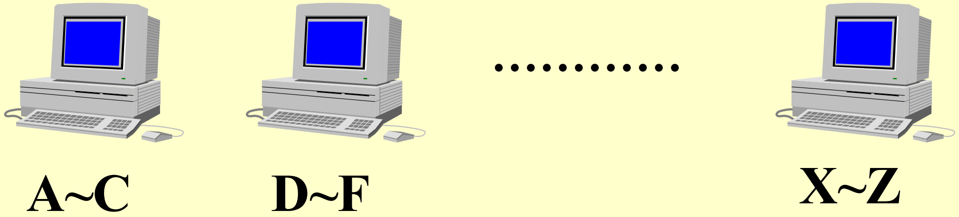
☞ **Solution 1: Term-partitioned index**



Distributed indexing (for web-scale indexing — don’t try this at home!)

—— Each node contains index of a subset of collection

☞ **Solution 1: Term-partitioned index**

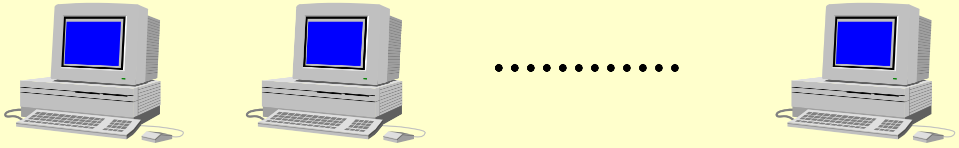


☞ **Solution 2: Document-partitioned index**

Distributed indexing (for web-scale indexing — don't try this at home!)

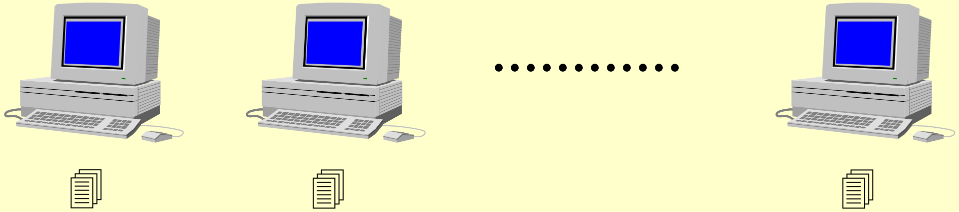
—— Each node contains index of a subset of collection

☞ **Solution 1: Term-partitioned index**



A~C **D~F** **X~Z**

☞ **Solution 2: Document-partitioned index**



1~ 10000 **10001~ 20000** **90001~ 100000**

Dynamic indexing

Dynamic indexing

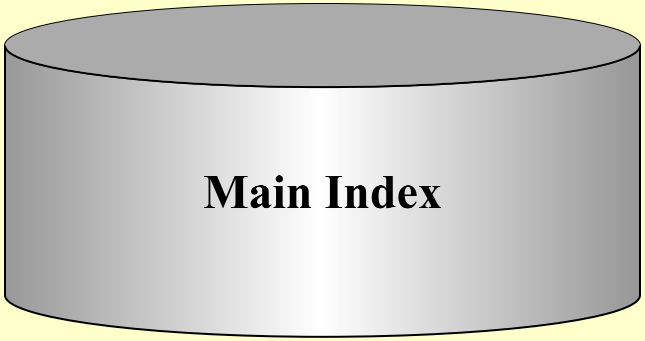
☞ Docs come in over time

- postings updates for terms already in dictionary
- new terms added to dictionary

☞ Docs get deleted

Dynamic indexing

- ☞ Docs come in over time
  - - postings updates for terms already in dictionary
  - - new terms added to dictionary
- ☞ Docs get deleted

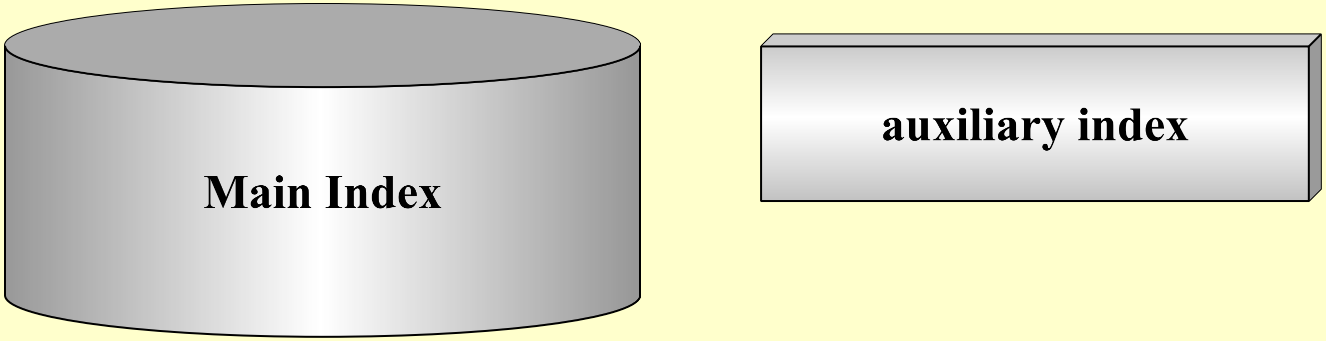


Dynamic indexing

☞ Docs come in over time

- postings updates for terms already in dictionary
- new terms added to dictionary

☞ Docs get deleted

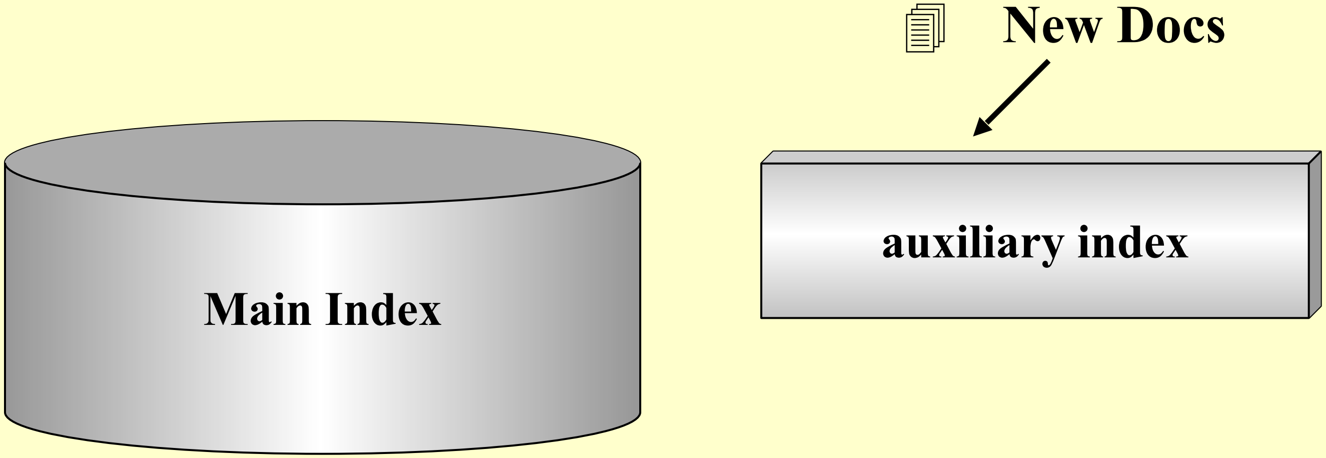


Dynamic indexing

☞ Docs come in over time

- postings updates for terms already in dictionary
- new terms added to dictionary

☞ Docs get deleted



New Docs

auxiliary index

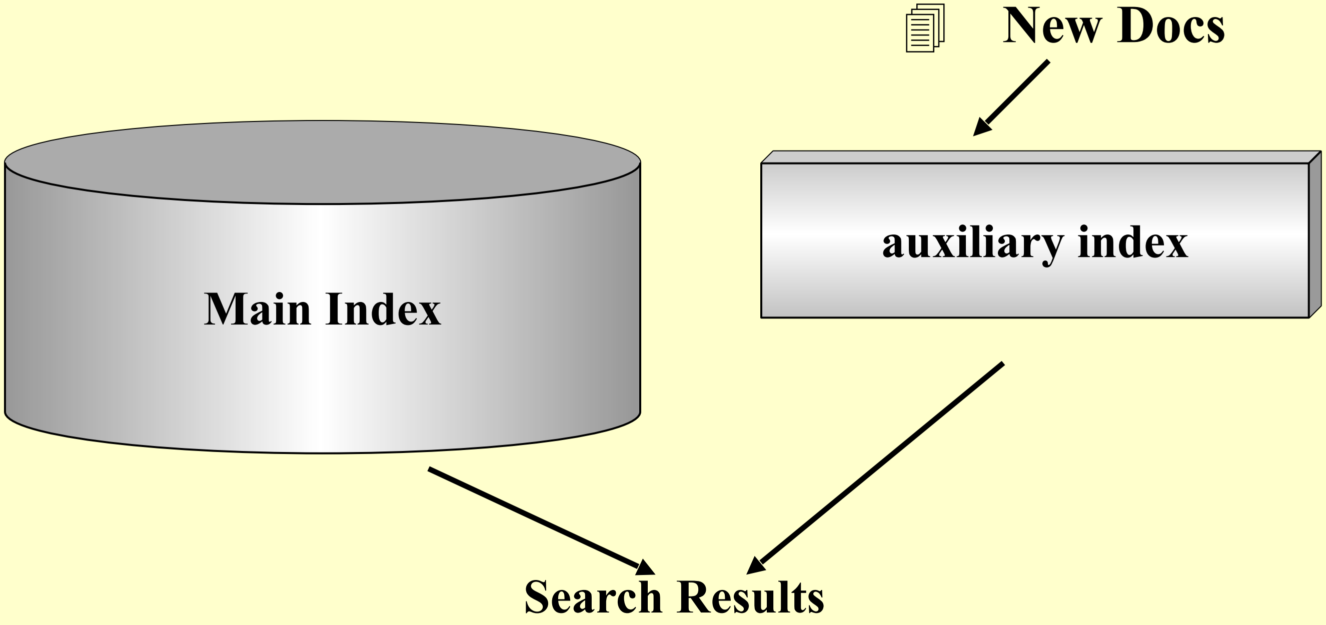
Main Index

Dynamic indexing

☞ Docs come in over time

- postings updates for terms already in dictionary
- new terms added to dictionary

☞ Docs get deleted

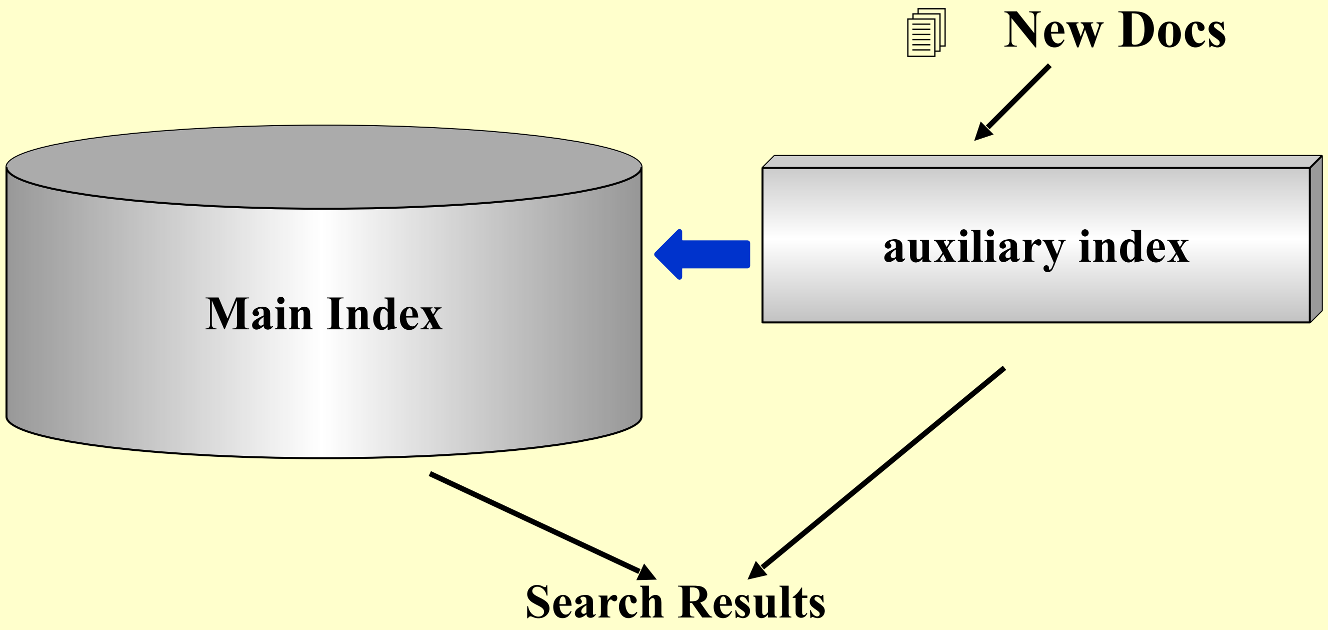


Dynamic indexing

☞ Docs come in over time

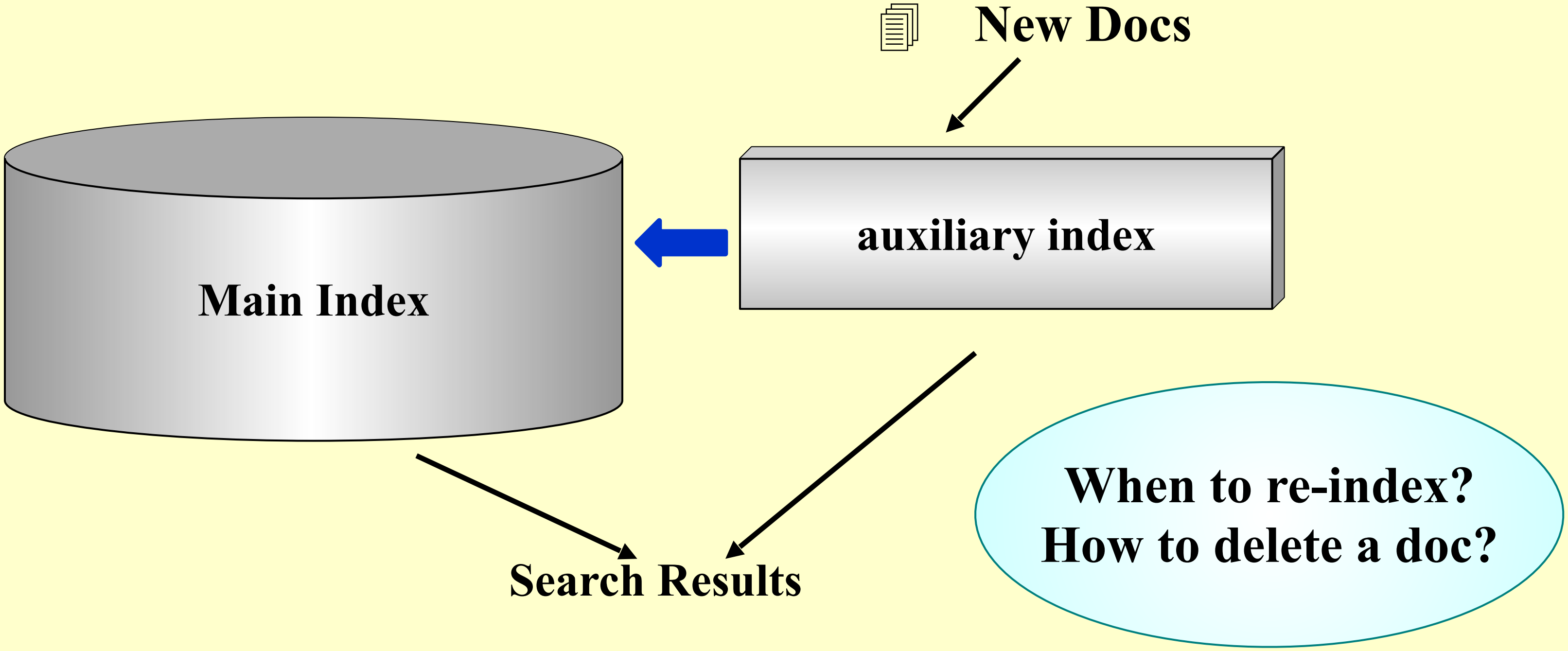
- postings updates for terms already in dictionary
- new terms added to dictionary

☞ Docs get deleted

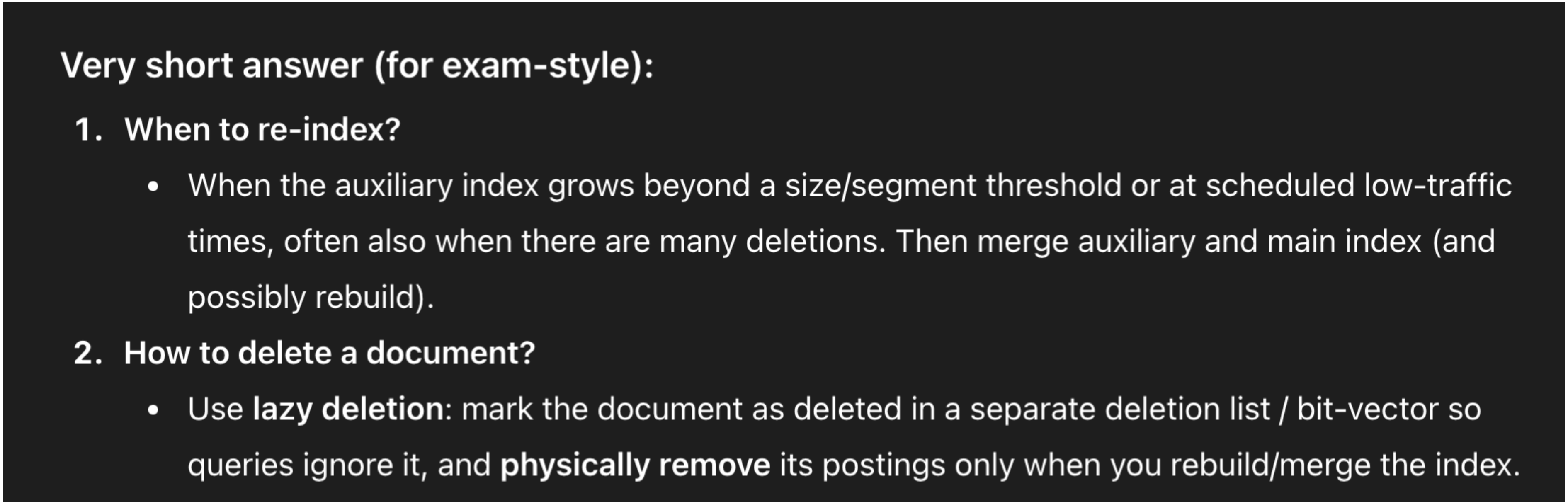


Dynamic indexing

- ☞ Docs come in over time
  - postings updates for terms already in dictionary
  - new terms added to dictionary
- ☞ Docs get deleted



When to re-index?
How to delete a doc?



Very short answer (for exam-style):

1. **When to re-index?**
   - When the auxiliary index grows beyond a size/segment threshold or at scheduled low-traffic times, often also when there are many deletions. Then merge auxiliary and main index (and possibly rebuild).
2. **How to delete a document?**
   - Use **lazy deletion**: mark the document as deleted in a separate deletion list / bit-vector so queries ignore it, and **physically remove** its postings only when you rebuild/merge the index.

Compression

Compression

Term
a
arrive
damage
deliver
fire
gold
of
in
shipment
silver
truck

Compression

Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck

Compression

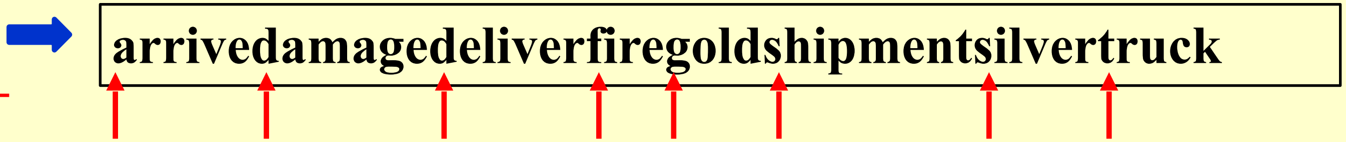
Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck

→

arrivedamagedeliverfiregoldshipmentsilvertruck

Compression

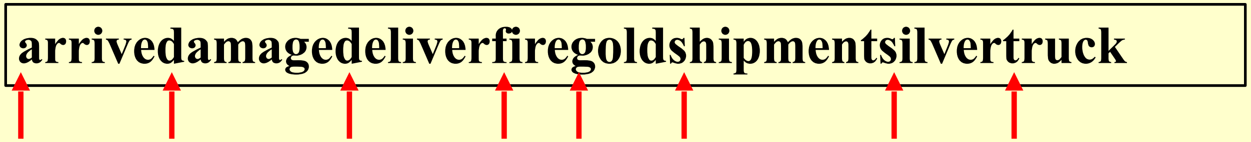
Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck



arrivedamagedeliverfiregoldshipmentsilvertruck

Compression

Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck

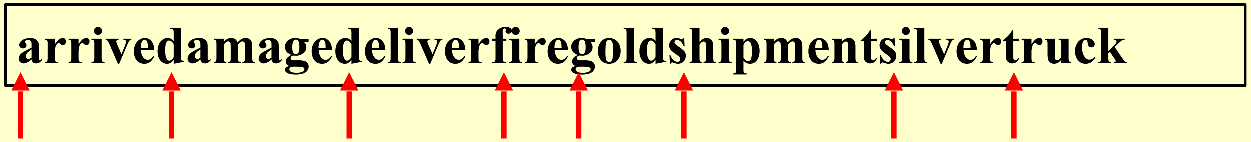


arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:

Compression

Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck



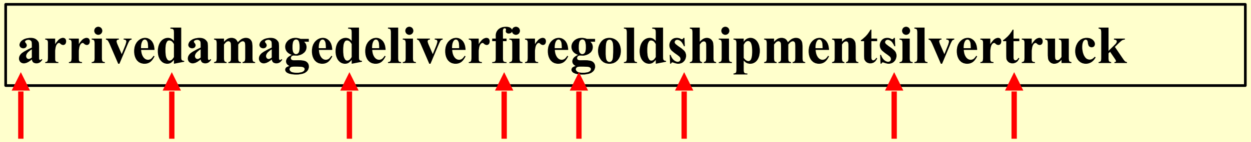
arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:

computer → 2, 15, 47, …, 58879, 58890, …

Compression

Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck



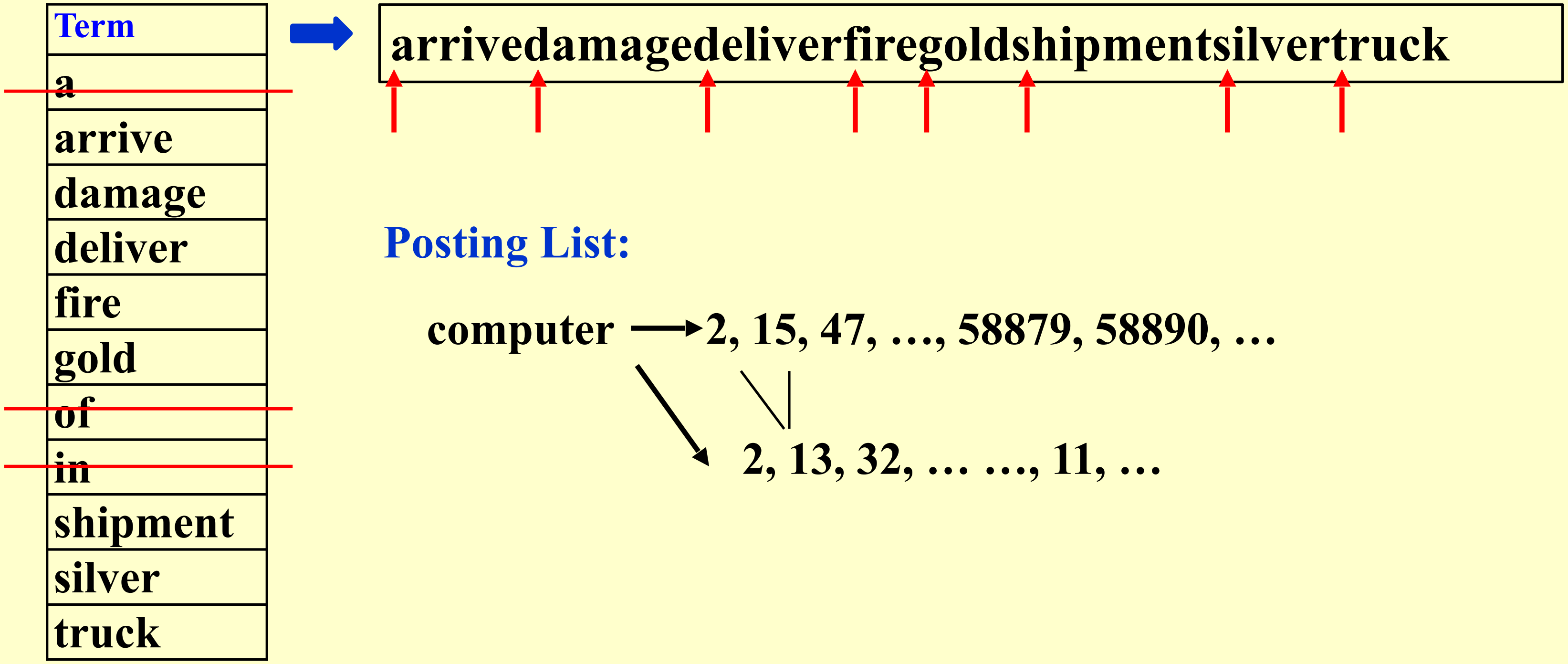
arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:

computer → 2, 15, 47, …, 58879, 58890, …

↘ 2, 13, 32, … …, 11, …

Compression



Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck

arrivedamagedeliverfiregoldshipmentsilvertruck

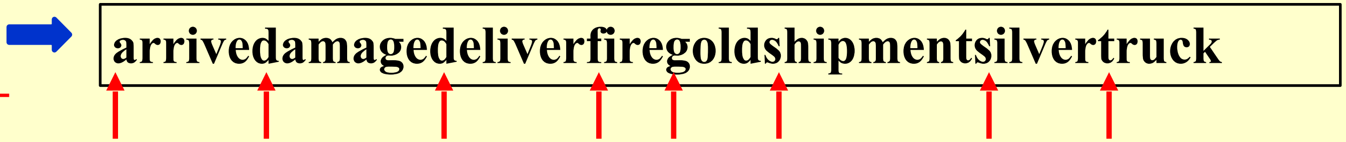
Posting List:

computer → 2, 15, 47, …, 58879, 58890, …

2, 13, 32, … …, 11, …

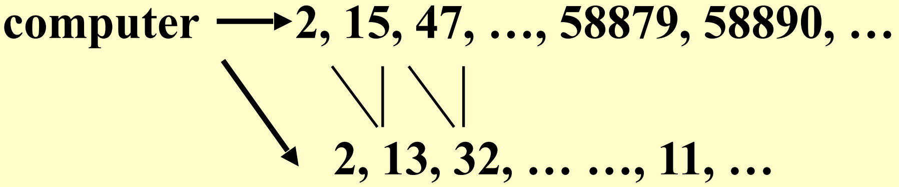
Compression

Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck



arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:

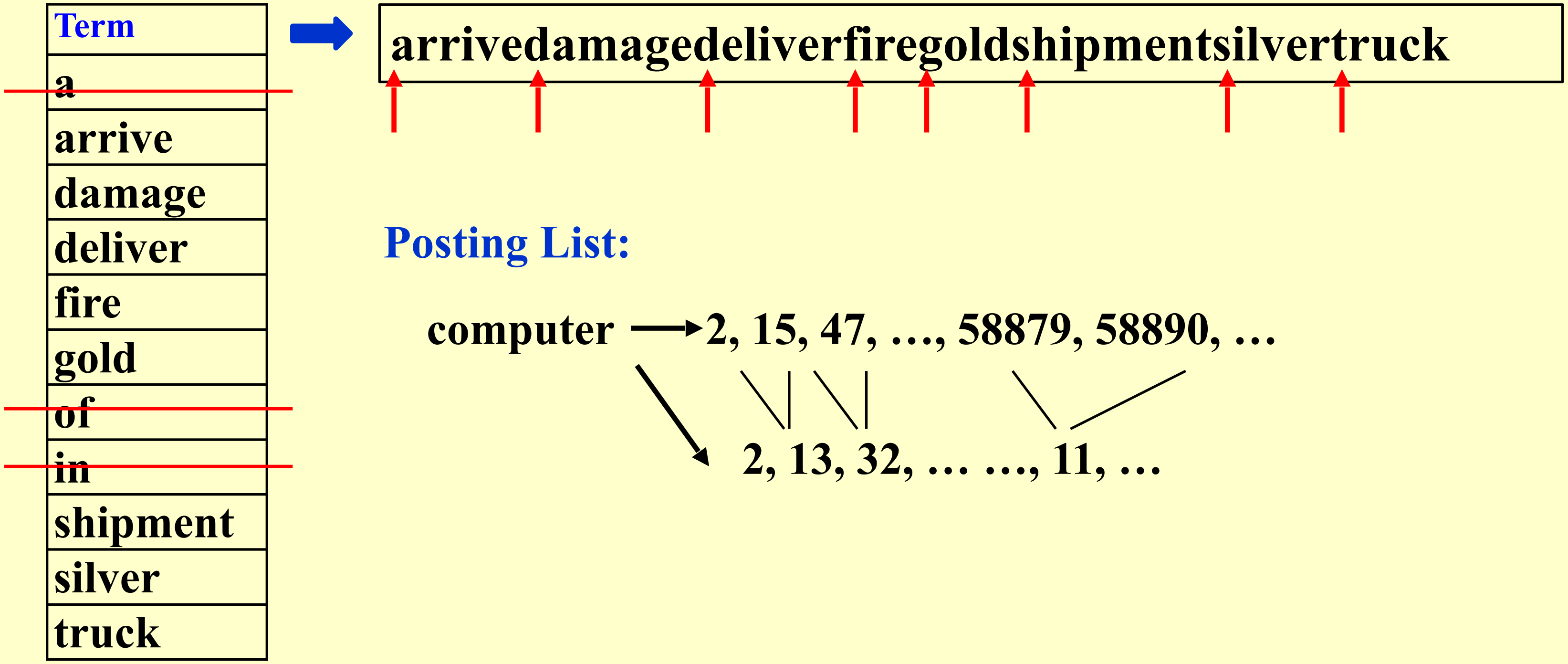


computer → 2, 15, 47, …, 58879, 58890, …

2, 13, 32, … …, 11, …

Inverted File Index

Compression



Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck

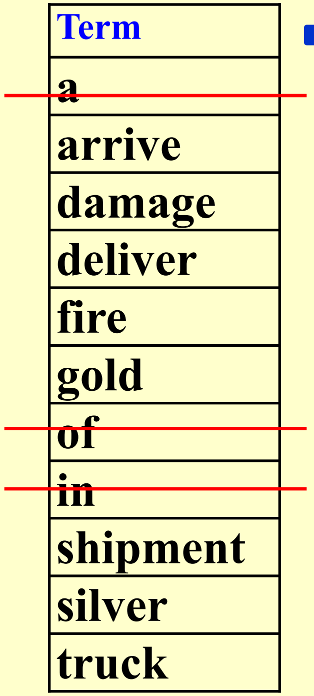
arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:

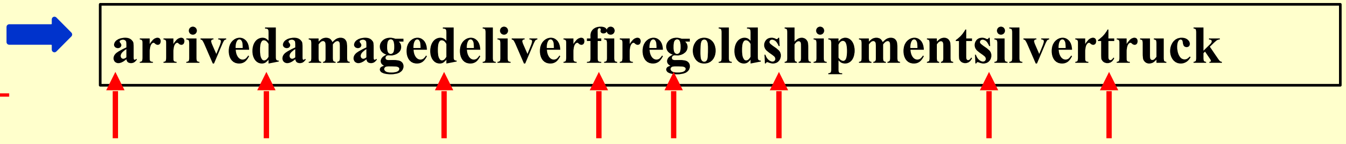
computer → 2, 15, 47, …, 58879, 58890, …

2, 13, 32, … …, 11, …

Compression

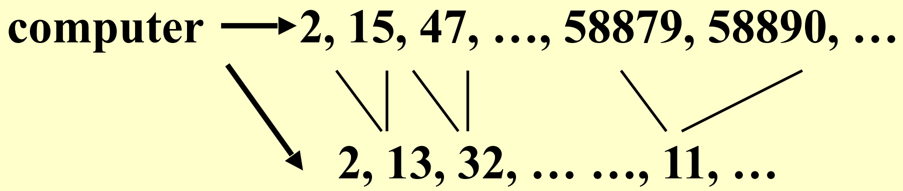


Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck



arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:



computer → 2, 15, 47, …, 58879, 58890, …

2, 13, 32, … …, 11, …

☝ Most gaps can be encoded with far fewer than 20 bits

Inverted File Index

Thresholding

Thresholding 阈值 **!=** 阀值

Thresholding

☞ **Document:** **only retrieve the top *x* documents where the documents are ranked by weight**

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

☞ **Query:** Sort the query terms by their frequency in ascending order; search according to only some percentage of the original query terms

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

☞ **Query:** Sort the query terms by their frequency in ascending order; search according to only some percentage of the original query terms

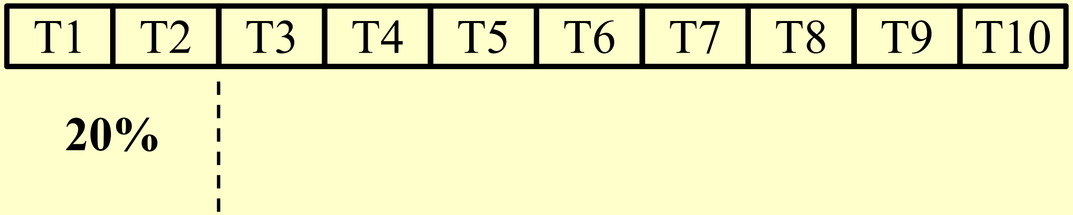
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

☞ **Query:** Sort the query terms by their frequency in ascending order; search according to only some percentage of the original query terms



T1	T2	T3	T4	T5	T6	T7	T8	T9	T10

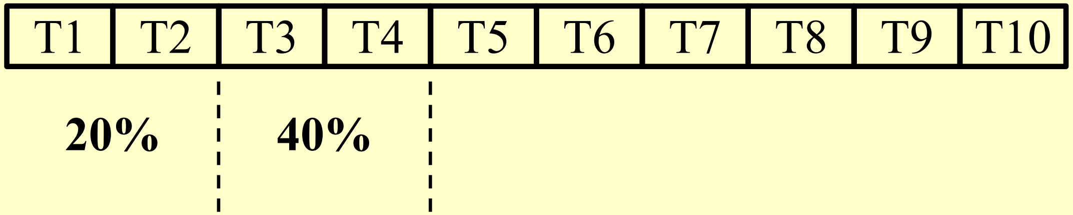
20%

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

☞ **Query:** Sort the query terms by their frequency in ascending order; search according to only some percentage of the original query terms



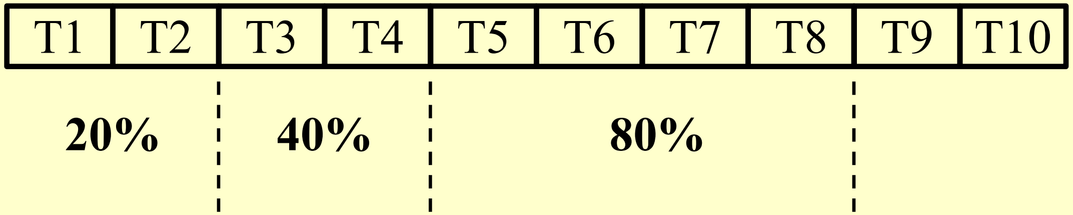
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

☞ **Query:** Sort the query terms by their frequency in ascending order; search according to only some percentage of the original query terms



T1	T2	T3	T4	T5	T6	T7	T8	T9	T10

Measures for a search engine

Measures for a search engine

☞ How fast does it index

- Number of documents/hour

☞ How fast does it search

- Latency as a function of index size

☞ Expressiveness of query language

- Ability to express complex information needs
- Speed on complex queries

Measures for a search engine

☞ How fast does it index

- Number of documents/hour

☞ How fast does it search

- Latency as a function of index size

☞ Expressiveness of query language

- Ability to express complex information needs
- Speed on complex queries

☞ **User happiness**

Measures for a search engine

☞ **How fast does it index**

- Number of documents/hour

☞ **How fast does it search**

- Latency as a function of index size

☞ **Expressiveness of query language**

- Ability to express complex information needs

- Speed on complex queries

☞ **User happiness ?**

Measures for a search engine

☞ How fast does it index

- Number of documents/hour

☞ How fast does it search

- Latency as a function of index size

☞ Expressiveness of query language

- Ability to express complex information needs
- Speed on complex queries

☞ **User happiness ?**

- Data Retrieval Performance Evaluation (after establishing correctness)
  > Response time
  > Index space
- Information Retrieval Performance Evaluation
  > + How *relevant* is the answer set?

Relevance **measurement requires 3 elements:**

1. **A benchmark document collection**
2. **A benchmark suite of queries**
3. **A binary assessment of either Relevant or Irrelevant for each query-doc pair**

Relevance **measurement requires 3 elements:**

1. **A benchmark document collection**
2. **A benchmark suite of queries**
3. **A binary assessment of either <u>Relevant</u> or <u>Irrelevant</u> for each query-doc pair**

	Relevant	**Irrelevant**
Retrieved	R_R	I_R
Not Retrieved	R_N	I_N

Relevance **measurement requires 3 elements:**

1. **A benchmark document collection**
2. **A benchmark suite of queries**
3. **A binary assessment of either Relevant or Irrelevant for each query-doc pair**

	Relevant	**Irrelevant**
Retrieved	R_R	I_R
Not Retrieved	R_N	I_N

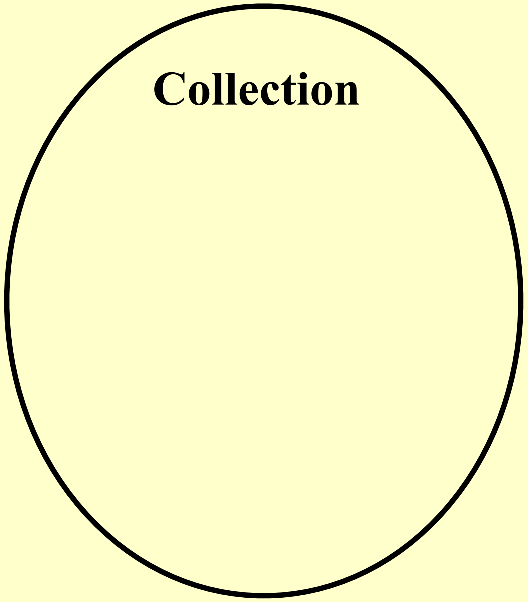
Precision $P = R_R / (R_R + I_R)$

Recall $R = R_R / (R_R + R_N)$

Relevance **measurement requires 3 elements:**

1. **A benchmark document collection**
2. **A benchmark suite of queries**
3. **A binary assessment of either Relevant or Irrelevant for each query-doc pair**

	Relevant	Irrelevant
Retrieved	R_R	I_R
Not Retrieved	R_N	I_N



Precision $P = R_R / (R_R + I_R)$

Recall $R = R_R / (R_R + R_N)$

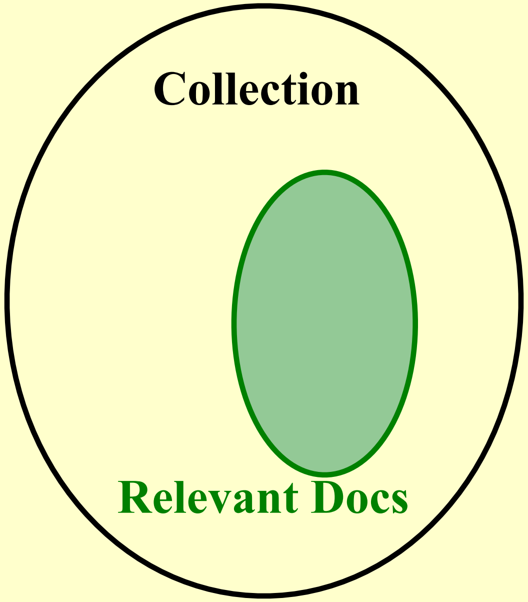
Relevance measurement requires 3 elements:

1. A benchmark document collection
2. A benchmark suite of queries
3. A binary assessment of either Relevant or Irrelevant for each query-doc pair

	Relevant	Irrelevant
Retrieved	R_R	I_R
Not Retrieved	R_N	I_N

Precision $P = R_R / (R_R + I_R)$

Recall $R = R_R / (R_R + R_N)$



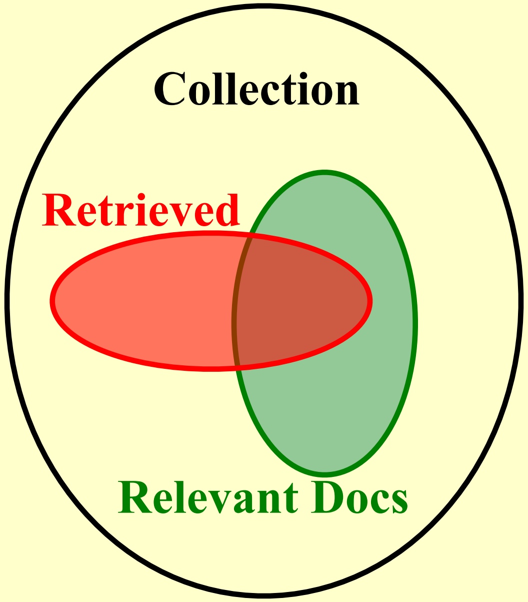
Relevance measurement requires 3 elements:

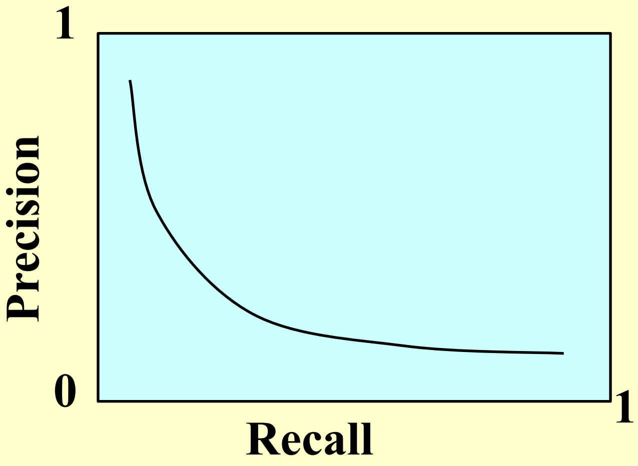
1. A benchmark document collection
2. A benchmark suite of queries
3. A binary assessment of either Relevant or Irrelevant for each query-doc pair

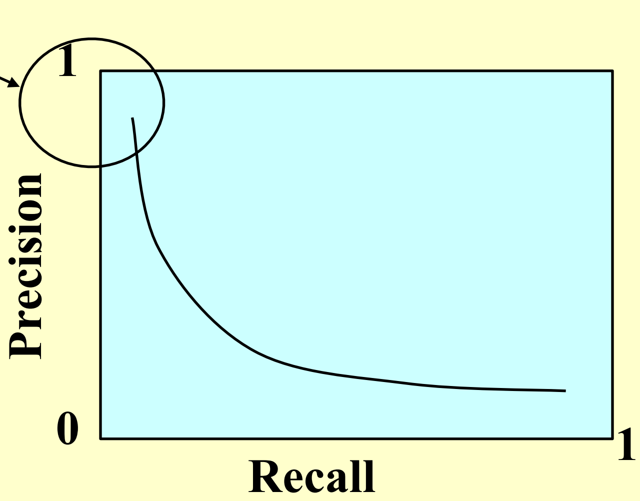
	Relevant	Irrelevant
Retrieved	R_R	I_R
Not Retrieved	R_N	I_N

Precision $P = R_R / (R_R + I_R)$

Recall $R = R_R / (R_R + R_N)$

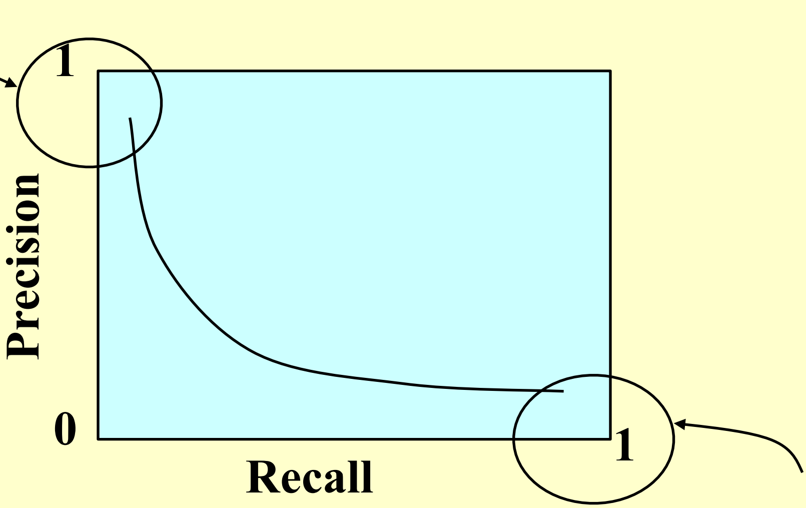



1
0
1
Precision
Recall



Returns relevant documents but misses many useful ones too

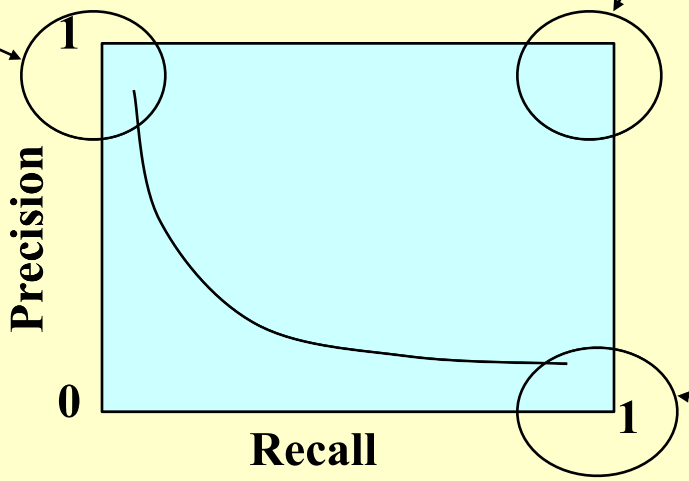
Returns relevant documents but misses many useful ones too



Returns most relevant documents but includes lot of junk

Returns relevant documents but misses many useful ones too

The ideal



Returns most relevant documents but includes lot of junk

Discussion 4:

How to improve the *relevancy* of search results?

Discussion 4:

How to improve the *relevancy* of search results?

- **Page Rank**
- **Semantic Web**
- **……**

Inverted File Index

- Inverted File Index

- Take-home messages

Take-Home Messages

- Inverted file index:
  - Mapping from items to posting lists of documents.
  - Sorted, frequency of appearance, positions.
  - Performance measures: precision, recall…

Thanks for your attention!
Discussions?

Reference

Stanford CS276: https://web.stanford.edu/class/cs276/

Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. Introduction to information retrieval. Cambridge University Press, 2008. Chap. 1, 4, 5, 6, 8

Zobel, Justin, and Alistair Moffat. "Inverted files for text search engines." ACM computing surveys, 2006.